AF

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 120 709 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
01.08.2001 Bulletin 2001/31

(51) Int Cl.[7]: **G06F 9/445**

(21) Application number: **01101820.7**

(22) Date of filing: **26.01.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **28.01.2000 JP 2000019997**

(71) Applicant: **NEC CORPORATION**
Tokyo (JP)

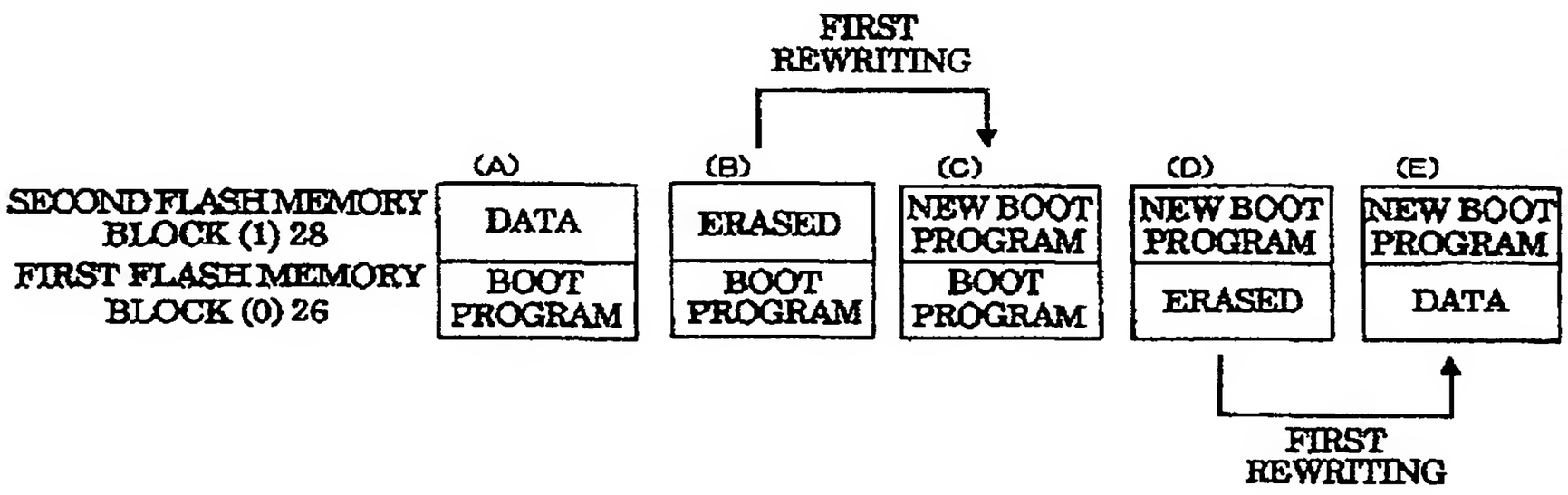
(72) Inventor: **Takahashi, Shuji,**
NEC IC Microcomputer Systems.
Kawasaki-shi, Kanagawa (JP)

(74) Representative: **Glawe, Delfs, Moll & Partner**
Patentanwälte
Postfach 26 01 62
80058 München (DE)

(54) Method of rewriting a boot program in a flash micro-computer

(57) A method of rewriting a boot program in a memory device including at least first and second memory blocks to which a boot program can be electrically written and from which a boot program can be electrically erased, includes the steps of (a) erasing data stored in the first memory block (28) to which a new boot program is to be written, (b) writing a new boot program into the first memory block (28), (c) changing an address of the first memory block (28) such that the first memory block (28) can be recognized as a boot block, and (d) erasing a boot program out of the second memory block (26).

FIG. 5



EP 1 120 709 A2

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0001] The invention relates to a method of rewriting a boot program stored in a memory device into a new boot program, an apparatus of doing the same, and a recording medium storing a program therein for causing a computer to carry out the method or to act as the apparatus.

DESCRIPTION OF THE PRIOR ART

[0002] There have been suggested a lot of methods of rewriting a program in a memory device, in particular, in a flash memory device.

[0003] For instance, Japanese Unexamined Patent Publication No. 11-96779 has suggested a method of rewriting a program in a flash memory device which method is capable of reducing the number of rewriting a program.

[0004] In the method, a header area and a program data area are arranged in each of blocks in a physical address associated with a flash memory device, and header areas and program data areas of the blocks are arranged in series in a logic address associated with a processor. When data stored in a program data area N is to be rewritten, it is usually necessary to rewrite a block in the program data area N and a block in the associated header area. However, the program data area N and the associated header area are located in the same block in the physical address of the flash memory device. Accordingly, only the block is rewritten.

[0005] Japanese Unexamined Patent Publication No. 11-316687, which is based on United States patent application No. 019178 filed on February 5, 1999 and assigned to Compaq Computer Corporation, has suggested an apparatus for compensating for system error occurring while the system is in a start-up process.

[0006] The suggested apparatus is comprised of a read only memory (ROM), a non-volatile random access memory (RAM), and a central processing unit (CPU). The read only memory stores a plurality of bootable flash images, and includes a non-programmable boot block. The non-volatile random access memory stores status data indicative of which flash image was selected. The central processing unit tests as to whether the selected flash image is perfect. If perfect, the central processing unit carries out the selected flash image, and if not, the central processing unit newly selects another flash image. The central processing unit tests as to whether the thus newly selected flash image is perfect, and carries out the selected flash image only when it is perfect. In accordance with the apparatus, since an imperfect flash image is not carried out, it would be possible to automatically recover the system.

[0007] Japanese Unexamined Patent Publication No. 6-266552, which is based on United States patent application No. 695952 filed on May 6, 1992 and assigned to Intel Corporation, has suggested a computer system which is capable of dynamically modifying or updating a part of code or data stored in a non-volatile memory device, without detaching any parts such as a cover from the computer system.

[0008] Specifically, the computer system is comprised of a processor carrying out a process logic, a non-volatile memory device coupled to the processor and storing data and operating system process logic therein, means coupled to the processor for reading out what is stored in the non-volatile memory device, means coupled to the processor for erasing what is stored in the non-volatile memory device, means coupled to the processor for programming the non-volatile memory device, and means for updating operating system process logic which the operating system process logic is being carried out.

[0009] Japanese Unexamined Patent Publication No. 10-177527 has suggested a method of updating data in a flash memory which method avoids data from being erased or from being changed into different data, even if power supply is stopped during data is being updated.

[0010] Specifically, the method updates data stored in a flash memory from which data can be erased block by block, and into which data can be written byte by byte. In the method, a flash memory is divided into two blocks. A unit of data is alternately written into the two blocks. When data is to be read out of the two blocks, a latest data unit among data units properly written into the blocks is read out.

[0011] Japanese Unexamined Patent Publication No. 8-280679 has suggested a program memory device in which a program can be readily rewritten.

[0012] The program memory device is comprised of a flash memory storing various programs and including a plurality of areas into which data can be electrically written and from which data can be electrically erased, a connector arranged externally of the program memory device operating in accordance with a program, and electrically connecting the program memory device to a memory storing a new program to be newly stored into the program memory device, first means for selecting one of a program of the flash memory and the new program stored in the memory, second means for storing the new program into a non-used area in the flash memory, and assigning an executable address to the new program, and a primary memory storing data used for converting the address to be carried out in the second means, into an address of a program stored in the flash memory, data about a degree to which the flash memory is used, and data about a total number of rewriting the flash memory.

[0013] If electric power supply to a computer is stopped for some reasons while a boot program stored in the computer is being written, it would be impossible

to restart the boot program.

[0014] Japanese Unexamined Patent Publication No. 8-255084, which is based on United States patent application No. 375,095 filed on January 18, 1995 and assigned to Hewlett Packard Company, has suggested a method for preventing such a trouble.

[0015] Figs. 1 and 2 are flow charts of the method suggested in Japanese Unexamined Patent Publication No. 8-255084.

[0016] Hereinbelow is explained the method in the assumption that a boot program stored in a primary boot block 0 is to be written into a new boot program.

[0017] As illustrated in Fig. 2-(A), in an initial state, a boot program is stored in a primary boot block 0, and data is stored in a substitution block 1.

[0018] First, as illustrated in Fig. 2-(B), data stored in the substitution block 1 is erased in step 300.

[0019] Then, as illustrated in Fig. 2-(C), a boot program stored in the primary boot block 0 is copied into the substitution block 1 in step 310.

[0020] Then, a non-volatile memory block circuit (not illustrated) changes an address of the substitution block 1 to such an address that the substitution block 1 can be recognized as a boot block, in step 320.

[0021] Then, as illustrated in Fig. 2-(D), a boot program stored in the primary boot block 0 is erased in step 330.

[0022] Then, as illustrated in Fig. 2-(E), a new boot program is written into the primary boot block 0 in step 340.

[0023] Then, the above-mentioned non-volatile memory block circuit (not illustrated) changes an address of the primary boot block 0 to such an address that the primary boot block 0 into which a new boot program has been stored can be recognized as a boot block, in step 350.

[0024] Then, as illustrated in Fig. 2-(F), the boot program which has been copied from the primary boot block 0 is erased from the substitution block 1, in step 360.

[0025] Thus, the substitution block 1 which is now empty can be used as an area for storing data therein. Hence, as illustrated in Fig. 2-(G), data is stored into the substitution block 1.

[0026] If a boot program has been properly rewritten in the primary boot block 0, the primary boot block 0 absolutely becomes a boot area, as illustrated in Fig. 2-(F). Accordingly, judgment as to whether a boot program has been properly rewritten is made by detecting whether the primary boot block 0 becomes a boot area. Hence, in the conventional method, a boot program has to be copied to the substitution block 1 from the primary boot block 0, as having been explained above, resulting in that extra stress is exerted on a flash memory of the substitution block 1.

[0027] In addition, in the conventional method having been explained with reference to Figs. 1 and 2, it is necessary to copy a boot program stored in the primary boot block 0 into the substitution block 1, and carry out an erasion/writing operation twice to the substitution block 1 in order to erase the boot program copied into the substitution block 1 and use the substitution block 1 as an area for storing data therein.

[0028] Specifically, the first rewriting is carried out when the status illustrated in Fig. 2-(B) is transferred to the status illustrated in Fig. 2-(C), and the second rewriting is carried out when the status illustrated in Fig. 2-(F) is transferred to the status illustrated in Fig. 2-(G).

[0029] The method or apparatus suggested in the above-mentioned Japanese Unexamined Patent Publications Nos. 11-96779, 11-316687, 6-266552, 10-177527 and 8-280679 has the step of copying a program to a memory block acting as a substitution block, resulting in exertion of extra stress on a flash memory of a memory block acting as a substitution block.

SUMMARY OF THE INVENTION

[0030] In view of the above-mentioned problems in the conventional methods and apparatuses, it is an object of the present invention to provide a method of rewriting a boot program which method makes it no longer necessary to copy a boot program from a memory block to another memory block when a boot program stored in a memory device is to be rewritten.

[0031] It is also an object of the present invention to provide a computer which is capable of doing the same.

[0032] In one aspect of the present invention, there is provided a method of rewriting a boot program in a memory device including at least first and second memory blocks to which a boot program can be electrically written and from which a boot program can be electrically erased, the method including the steps of (a) erasing data stored in the first memory block to which a new boot program is to be written, (b) writing a new boot program into the first memory block, (c) changing an address of the first memory block such that the first memory block can be recognized as a boot block, and (d) erasing a boot program out of the second memory block.

[0033] In accordance with the above-mentioned method, it is no longer necessary to copy a boot program stored in a memory block (the second memory block) into another memory block acting as a substitution block (the first memory block), unlike the above-mentioned conventional method, and it is possible to write a new boot program directly into the first memory block acting as a substitution block. Hence, an erasion/writing operation of a boot program to the first memory block acting as a substitution block is carried out only once, whereas the same operation had to be carried out twice in the conventional method. The method makes it possible to reduce the number of erasion/writing of a boot program to the first memory block acting as a substitution block, ensuring reduction in stress to be exerted on a flash memory of the first memory block, and hence, lengthening a lifetime of the flash memory.

[0034] It is preferable that the method further includes

the step of writing data about the new boot program into a storage area associated with the first memory block.

[0035] This additional step ensures the first memory block to be recognized as a boot block area.

[0036] It is preferable that the method further includes the step of judging that a boot program was not properly rewritten, if a boot program is stored in both the first and second memory blocks.

[0037] It is preferable that the method further includes the step of judging that a boot program was not properly rewritten, if data about the new boot program is stored in storage areas associated with both the first and second memory blocks.

[0038] It is preferable that the memory device includes three or more memory blocks, and two memory blocks among the memory blocks are used.

[0039] The method in accordance with the present invention can be applied to a memory device including three or more memory blocks as well as a memory device including two memory blocks. As the memory device includes the greater number of memory blocks, the memory device could have a longer lifetime.

[0040] There may be selected a flash memory or a read only memory (ROM) as the memory device.

[0041] In another aspect of the present invention, there is provided a computer including (a) a memory device including at least two memory blocks, (b) a controller which controls writing a boot program into the memory blocks and erasing a boot program from the memory blocks, and (c) an address changer which changes an address of the memory blocks, characterized in that the controller erases data stored in a first memory block among the memory blocks, writes a new boot program into the first memory block, and erases a boot program stored in a second memory block among the memory blocks, and the address changer changes an address of the first memory block such that the first memory block can be recognized as a boot block.

[0042] In accordance with the above-mentioned computer, it is no longer necessary to copy a boot program stored in a memory block (the second memory block) into another memory block acting as a substitution block (the first memory block), unlike the above-mentioned conventional apparatus, and it is possible to write a new boot program directly into the first memory block acting as a substitution block. Hence, an erasion/writing operation of a boot program to the first memory block acting as a substitution block is carried out only once, whereas the same operation had to be carried out twice in the conventional apparatus. The computer makes it possible to reduce the number of erasion/writing of a boot program to the first memory block acting as a substitution block, ensuring reduction in stress to be exerted on a flash memory of the first memory block, and hence, lengthening a lifetime of the flash memory.

[0043] It is preferable that the controller writes data about the new boot program into a storage area associated with the first memory block.

[0044] The controller ensures that the first memory block is recognized as a boot block area.

[0045] It is preferable that the controller judges that a boot program was not properly rewritten, if a boot program is stored in both the first and second memory blocks.

[0046] It is preferable that the controller judges that a boot program was not properly rewritten, if data about the new boot program is stored in storage areas associated with both the first and second memory blocks.

[0047] There may be selected a flash memory or a read only memory (ROM) as the memory device.

[0048] In still another aspect of the present invention, there is provided a recording medium readable by a computer, storing a program therein for causing a computer to carry out the above-mentioned method of rewriting a boot program in a memory device.

[0049] There is further provided a recording medium readable by a computer, storing a program therein for causing a computer to act as the above-mentioned computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0050] Fig. 1 is a flow chart showing respective steps of a conventional method of rewriting a program.

[0051] Fig. 2 is a flow chart showing how a primary boot block and a substitution block are rewritten in the conventional method illustrated in Fig. 1.

[0052] Fig. 3 is a block diagram of a flash microcomputer in which a method of rewriting a program in accordance with a preferred embodiment of the present invention is reduced into practice.

[0053] Fig. 4 is a flow chart showing an operation of the flash microcomputer illustrated in Fig. 3.

[0054] Fig. 5 is a flow chart showing how two memory blocks are rewritten in a flash memory constituting a part of the flash microcomputer illustrated in Fig. 3.

[0055] Fig. 6 is a block diagram showing status of the first and second flash memory blocks while a program is being rewritten.

[0056] Fig. 7 is a block diagram showing status of the first and second flash memory blocks while a program is being rewritten.

[0057] Fig. 8 is a block diagram showing status of the first and second flash memory blocks while a program is being rewritten.

[0058] Fig. 9 is a block diagram showing status of the first and second flash memory blocks while a program is being rewritten.

[0059] Fig. 10 is a block diagram showing status of the first and second flash memory blocks while a program is being rewritten.

[0060] Fig. 11 is a block diagram of an example of a circuit for changing an address, in the flash microcomputer illustrated in Fig. 3.

[0061] Fig. 12 is a block diagram of another example of a circuit for changing an address, in the flash micro-

computer illustrated in Fig. 3.

[0062] Fig. 13 illustrates examples of recording mediums in which a program for rewriting a program in a flash microcomputer is to be stored.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0063] Preferred embodiments in accordance with the present invention will be explained hereinbelow with reference to drawings.

[0064] Fig. 3 is a block diagram of a flash microcomputer in which a method of rewriting a program in accordance with an embodiment of the present invention is reduced into practice.

[0065] A flash microcomputer in accordance with the embodiment includes a flash memory 20.

[0066] The flash memory 20 includes a first flash memory block (0) 26, a second flash memory block (1) 28, a third flash memory block (2) 30, and a fourth flash memory block (n) 32. A boot program can be electrically written into the first to fourth flash memory blocks 26 to 32, and a boot program can be erased from the first to fourth flash memory blocks 26 to 32.

[0067] The flash memory 20 further includes a first storage area (0) 27 associated with the first flash memory block (0) 26, a second storage area (1) 29 associated with the second flash memory block (1) 28, a third storage area (2) 31 associated with the third flash memory block (2) 30, and a fourth storage area (n) 33 associated with the fourth flash memory block (n) 32. The first to fourth storage areas 27 to 33 store data about a boot program stored in the associated flash memory block 26 to 32.

[0068] The flash microcomputer 10 further includes a memory 22 storing a program used for controlling writing a boot program into the flash memory 20 and erasing a boot program from the flash memory 20, a flash memory control circuit 23 which carried out writing a boot program into the flash memory 20 and erasing a boot program from the flash memory 20 in accordance with the program stored in the memory 22, an address changing circuit 24 which changes addresses of the first to fourth flash memory blocks (0) 26 to (n) 32, a random access memory (RAM) 25, and a central processing unit (CPU) 21 which controls the random access memory 25 and the address changing circuit 24 in accordance with the program stored in the memory 22.

[0069] Figs. 4 and 5 are flow charts showing an operation of the flash microcomputer 10 illustrated in Fig. 3. Hereinbelow is explained an operation of the flash microcomputer 10 with reference to Figs. 4 and 5.

[0070] It is assumed in the present embodiment that the second flash memory block (1) 28 is used as a substitution block into which a new boot program is to be written, and a boot program stored in the first flash memory block (0) 26 is rewritten into a new boot program.

[0071] Before rewriting a boot program, as illustrated in Fig. 5-(A), the first flash memory block (0) 26 stores a boot program therein, and the second flash memory block (1) 28 stores data therein.

[0072] First, as illustrated in Fig. 5-(B), data is erased from the second flash memory block (1) 28 acting as a substitution block into which a new boot program is to be written, in step 100.

[0073] Fig. 6 illustrates status of both the first flash memory block (0) 26 and the second flash memory block (1) 28 at this stage. The first flash memory block (0) 26 still stores a boot program therein, and the second flash memory block (1) 28 is empty.

[0074] The first storage area (0) 27 associated with the first flash memory block (0) 26 stores program data 00h indicating that a boot program is stored in an associated flash memory block, and the second storage area (1) 29 associated with the second flash memory block (1) 28 stores program data FFh indicating that an associated flash memory block stores nothing, that is, an associated flash memory block is empty.

[0075] Then, as illustrated in Fig. 5-(C), a new boot program is written into the second flash memory block (1) 28 acting as a substitution block, in step 110.

[0076] Fig. 7 illustrates status of both the first flash memory block (0) 26 and the second flash memory block (1) 28 at this stage. The first flash memory block (0) 26 still stores a boot program therein, and the second flash memory block (1) 28 stores the new boot program.

[0077] The first storage area (0) 27 associated with the first flash memory block (0) 26 stores program data 00h indicating that a boot program is stored in an associated flash memory block, and the second storage area (1) 29 associated with the second flash memory block (1) 28 still stores program data FFh indicating that an associated flash memory block stores nothing.

[0078] Then, new program data is stored into the second storage area (1) 29 associated with the second flash memory block (1) 28 into which the new boot program has been written, in step 120.

[0079] Fig. 8 illustrates status of both the first flash memory block (0) 26 and the second flash memory block (1) 28 at this stage. The first flash memory block (0) 26 still stores a boot program therein, and the second flash memory block (1) 28 stores the new boot program.

[0080] The first storage area (0) 27 associated with the first flash memory block (0) 26 stores program data 00h indicating that a boot program is stored in an associated flash memory block, and the second storage area (1) 29 associated with the second flash memory block (1) 28 stores program data 01h indicating that an associated flash memory block stores another boot program.

[0081] Then, the central processing unit 21 causes the address changing circuit 24 to change an address of the second flash memory block (1) 28 such that the second flash memory block (1) 28 into which the new boot program has been written can be recognized as a boot block, in step 130.

[0082] Fig. 9 illustrates status of both the first flash

memory block (0) 26 and the second flash memory block (1) 28 at this stage. Similarly to Fig. 8, the first flash memory block (0) 26 still stores a boot program therein, and the second flash memory block (1) 28 stores the new boot program.

[0083] However, in the status illustrated in Fig. 9, an address of the second flash memory block (1) 28 into which the new boot program has been written is changed to "00000h□3FFFFh" from "40000h□7FFFFh", unlike an address of the second flash memory block (1) 28 in the status illustrated in Fig. 8. By changing the address, the second flash memory block (1) 28 into which the new boot program has been written is recognized as a boot block by the central processing unit 21.

[0084] In contrast, an address of the first flash memory block (0) 26 still storing the old boot program is changed to "40000h□7FFFFh" from "00000h□3FFFFh". By changing the address, the first flash memory block (0) 26 is not recognized as a boot block by the central processing unit 21.

[0085] Then, as illustrated in Fig. 5-(D), the boot program stored in the first flash memory block (0) 26 is erased, in step 140.

[0086] Fig. 10 illustrates status of both the first flash memory block (0) 26 and the second flash memory block (1) 28 at this stage. The first flash memory block (0) 26 is empty, and the second flash memory block (1) 28 stores the new boot program.

[0087] The first storage area (0) 27 associated with the first flash memory block (0) 26 stores program data FFh indicating that an associated flash memory block stores nothing, and the second storage area (1) 29 associated with the second flash memory block (1) 28 stores program data 01h indicating that an associated flash memory block stores a boot program.

[0088] Then, as illustrated in Fig. 5-(E), data is stored into the empty first flash memory block (0) 26.

[0089] Thus, the operation of rewriting a boot program in the flash microcomputer 10 is completed.

[0090] Herein, it is assumed that electric power supply is stopped in the status illustrated in Fig. 8, that is, when the first flash memory block (0) 26 stores an old boot program, and a new boot program is being written into the second flash memory block (1) 28.

[0091] Accordingly, when the flash microcomputer 10 restarts, the flash microcomputer 10 includes two boot programs. That is, both the first storage area (0) 27 and the second storage area (1) 29 store program data indicating that a boot program is stored in an associated flash memory block. Based on such program data stored in both the first storage area (0) 27 and the second storage area (1) 29, it is judged that a program rewriting operation has not been properly completed. Thus, the flash microcomputer 10 can judge that it is necessary to make recovery.

[0092] Figs. 11 and 12 illustrate examples of systems in which a boot block memory address is changed by the address changing circuit 24.

[0093] In the system illustrated in Fig. 11, an ordinary memory address is indicated in order to boot the first flash memory block (0) 26 at first. In the system illustrated in Fig. 12, a memory address is indicated in order to boot the second flash memory block (1) 28 at first.

[0094] A register 44 for designating a boot block can be set or reset bit by bit.

[0095] Fig. 11 illustrates a status in which the register 44 is in a reset condition. An input to XOR gate 42 is determined such that an upper address bit line 43 is not inverted. A lower address bit line 41 designates a memory 40 in a conventional manner.

[0096] Fig. 12 illustrates a status in which the register 44 is in a set condition. An input to XOR gate 42 is determined such that the upper address bit line 43 is inverted. Accordingly, the second flash memory block (1) 28 is recognized as a boot block.

[0097] In the conventional method having been explained with reference to Figs. 1 and 2, it is necessary to copy a boot program stored in the primary boot block 0 into the substitution block 1, as illustrated in Fig. 2-(C) and it is also necessary to carry out an erasion/writing operation twice to the substitution block 1 in order to erase the boot program copied into the substitution block 1 and use the substitution block 1 as an area for storing data therein, as illustrated in Fig. 2-(F) and 2-(G).

[0098] In accordance with the above-mentioned method, it is no longer necessary to copy a boot program stored in a primary boot block (the first flash memory block (0) 26) into the second flash memory block (1) 28 acting as a substitution block, unlike the conventional method. It is possible to write a new boot program directly into the second flash memory block (1) 28 acting as a substitution block, in place of copying a boot program to the second flash memory block (1) 28. Hence, an erasion/writing operation of a boot program to the second flash memory block (1) 28 acting as a substitution block is carried out only once, as illustrated in Fig. 5-(C), whereas the same operation had to be carried out twice in the conventional method.

[0099] As mentioned above, the method in accordance with the embodiment makes it possible to reduce the number of erasion/writing of a boot program to the second flash memory block (1) 28 acting as a substitution block, ensuring reduction in stress to be exerted on a flash memory of the second flash memory block (1) 28, and hence, lengthening a lifetime of the flash memory.

[0100] Though the program rewriting operation is carried out among the two flash memory blocks in the above-mentioned embodiment, the method in accordance the present invention can be applied to a flash memory including three or more flash memory blocks. A flash memory including the greater number of flash memory blocks would have a longer lifetime.

[0101] The method of rewriting a boot program is explained in connection with a flash memory in the above-

mentioned embodiment. However, it should be noted that the present invention is not to be limited to a flash memory, but can be applied to any device having a necessity of rewriting a program.

[0102] For instance, there may be used other ROMs such as EEPROM as well as a flash memory.

[0103] The flash microcomputer in accordance with the above-mentioned embodiment is designed to include the first to fourth storage areas 27, 29, 31 and 33 for storing boot program data therein, in the flash memory 20 which storage areas 27, 29, 31 and 33 are separate from user areas. However, the storage areas 27, 29, 31 and 33 may be arranged in user areas. It is not always necessary to prepare the first to fourth storage areas 27, 29, 31 and 33 separately from user areas.

[0104] The first to fourth storage areas 27, 29, 31 and 33 may be designed to have a greater capacity in order to store other data therein. For instance, the storage areas 27, 29, 31 and 33 may be used as areas associated with the first to fourth flash memory blocks 26 to 32 for controlling the number of rewriting a program.

[0105] The method of rewriting a program having been mentioned so far may be accomplished as a program including various commands, and be presented through a recording medium readable by a computer.

[0106] In the specification, the term "recording medium" means any medium which can record data therein. Examples of a recording medium are illustrated in Fig. 13.

[0107] The term "recording medium" includes, for instance, a disk-shaped recorder 401 such as CD-ROM (Compact Disk-ROM) or PD, a magnetic tape, MO (Magneto Optical Disk), DVD-ROM (Digital Video Disk-Read Only Memory), DVD-RAM (Digital Video Disk-Random Access Memory), a floppy disk 402, a memory chip 404 such as RAM (Random Access Memory) or ROM (Read Only Memory), EPROM (Erasable Programmable Read Only Memory), EEPROM (Electrically Erasable Programmable Read Only Memory), smart media (Registered Trade Mark), a flush memory, a rewritable card-type ROM 405 such as a compact flush card, a hard disk 403, and any other suitable means for storing a program therein.

[0108] A recording medium storing a program for accomplishing the above-mentioned apparatus may be accomplished by programming functions of the above-mentioned apparatuses with a programming language readable by a computer, and recording the program in a recording medium such as mentioned above.

[0109] A hard disc equipped in a server may be employed as a recording medium. It is also possible to accomplish the recording medium in accordance with the present invention by storing the above-mentioned computer program in such a recording medium as mentioned above, and reading the computer program by other computers through a network.

[0110] As a computer 400, there may be used a personal computer, a desk-top type computer, a note-book type computer, a mobile computer, a lap-top type computer, a pocket computer, a server computer, a client computer, a workstation, a host computer, a commercially available computer, and electronic exchanger, for instance.

Claims

1. A method of rewriting a boot program in a memory device including at least first and second memory blocks to which a boot program can be electrically written and from which a boot program can be electrically erased,

   the method comprising the steps of:

   (a) erasing data stored in the first memory block (28) to which a new boot program is to be written;
   (b) writing a new boot program into the first memory block (28);
   (c) changing an address of the first memory block (28) such that the first memory block (28) can be recognized as a boot block; and
   (d) erasing a boot program out of the second memory block (26).

2. The method as set forth in claim 1, further comprising the step of writing data about the new boot program into a storage area (29) associated with the first memory block (28).

3. The method as set forth in claim 1 or 2, further comprising the step of judging that a boot program was not properly rewritten, if a boot program is stored in both the first and second memory blocks (28, 26).

4. The method as set forth in claim 1 or 2, further comprising the step of judging that a boot program was not properly rewritten, if data about the new boot program is stored in storage areas (29, 27) associated with both the first and second memory blocks (28, 26).

5. The method as set forth in claim 1 or 2, wherein the memory device includes three or more memory blocks, and two memory blocks among the memory blocks are used.

6. A computer comprising:

   (a) a memory device (20) including at least two memory blocks (26, 28, 30, 32);
   (b) a controller (23) which controls writing a boot program into the memory blocks (26, 28, 30, 32) and erasing a boot program from the memory blocks (26, 28, 30, 32); and
   (c) an address changer (24) which changes an

address of the memory blocks (26, 28, 30, 32),

characterized in that

the controller (23) erases data stored in a first memory block (28) among the memory blocks, writes a new boot program into the first memory block (28), and erases a boot program stored in a second memory block (26) among the memory blocks, and
the address changer (24) changes an address of the first memory block (28) such that the first memory block (28) can be recognized as a boot block.

7. The computer as set forth in claim 6, wherein the controller (23) writes data about the new boot program into a storage area (29) associated with the first memory block (28).

8. The computer as set forth in claim 6 or 7, wherein the controller (23) judges that a boot program was not properly rewritten, if a boot program is stored in both the first and second memory blocks (28, 26).

9. The computer as set forth in claim 6 or 7, wherein the controller (23) judges that a boot program was not properly rewritten, if data about the new boot program is stored in storage areas associated with both the first and second memory blocks (28, 26).

10. The computer as set forth in claim 6 or 7, wherein the memory device (20) is a flash memory.

11. The computer as set forth in claim 6 or 7, wherein the memory device (20) is a read only memory (ROM).

12. A recording medium readable by a computer, storing a program therein for causing a computer to carry out one the method defined in any one of claims 1 to 5.

13. A recording medium readable by a computer, storing a program therein for causing a computer to act as the computer defined in any one of claims 6 to 11.

FIG. 1
PRIOR ART

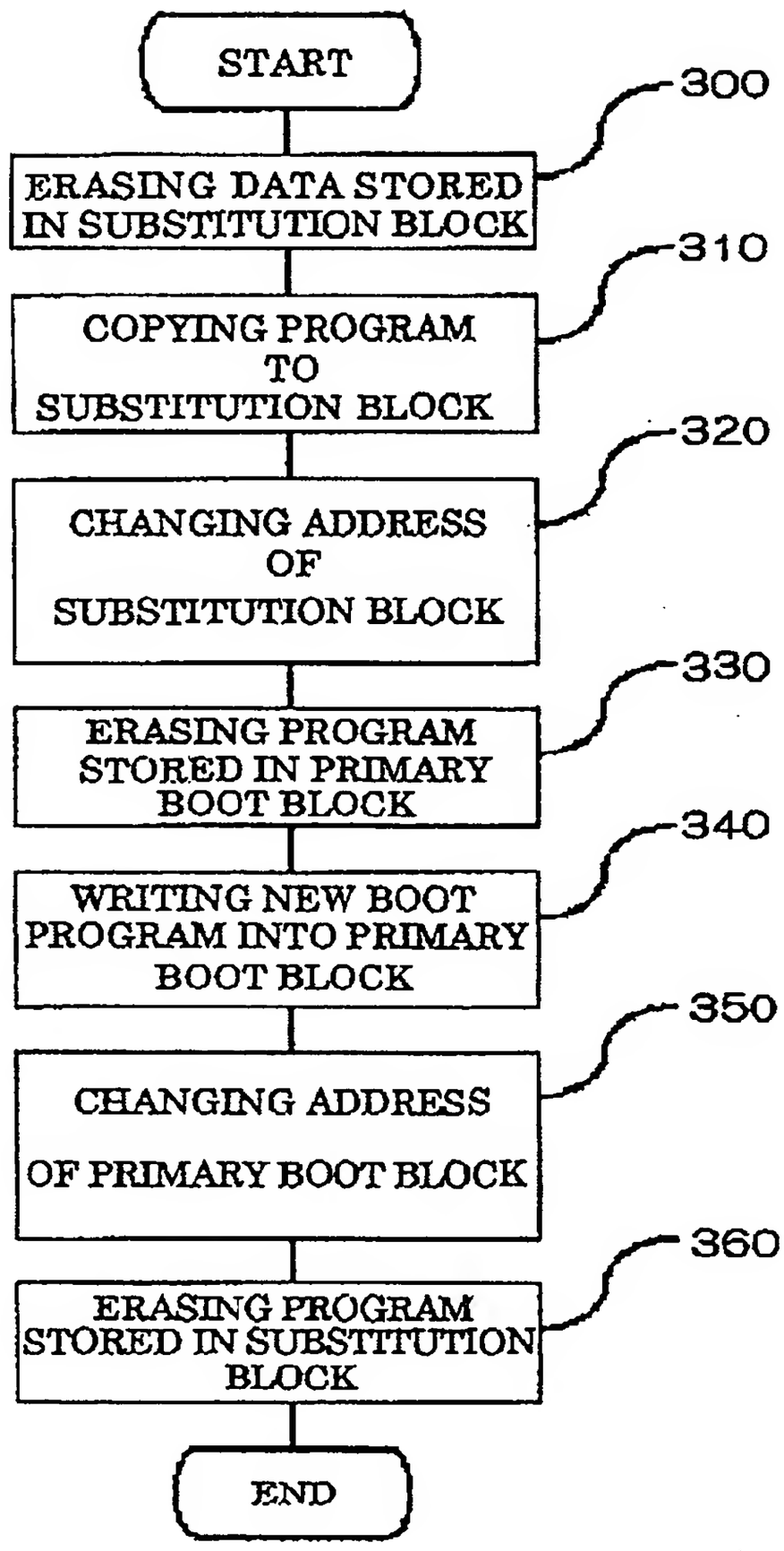


FIG. 2
PRIOR ART

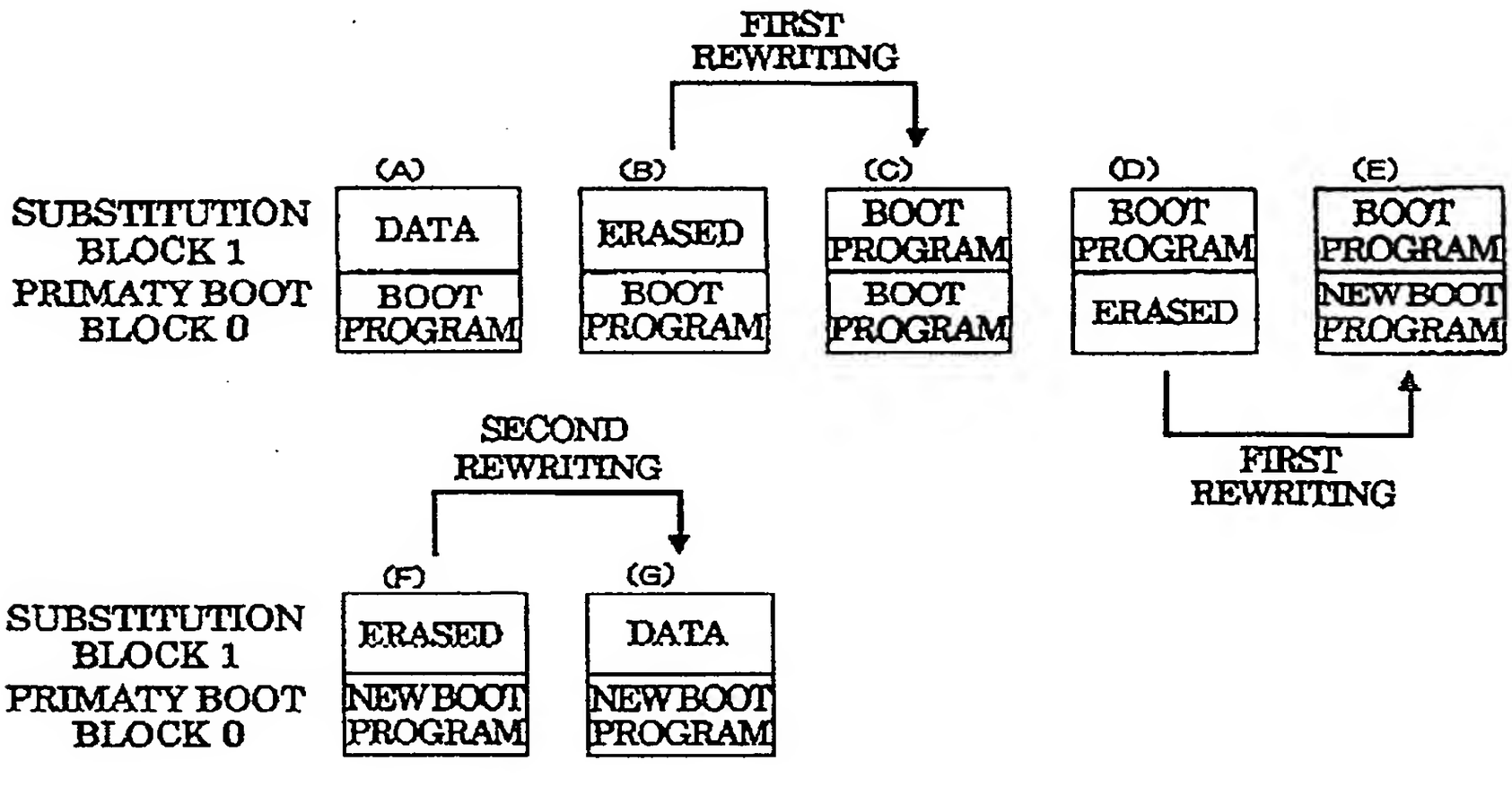


FIG. 3

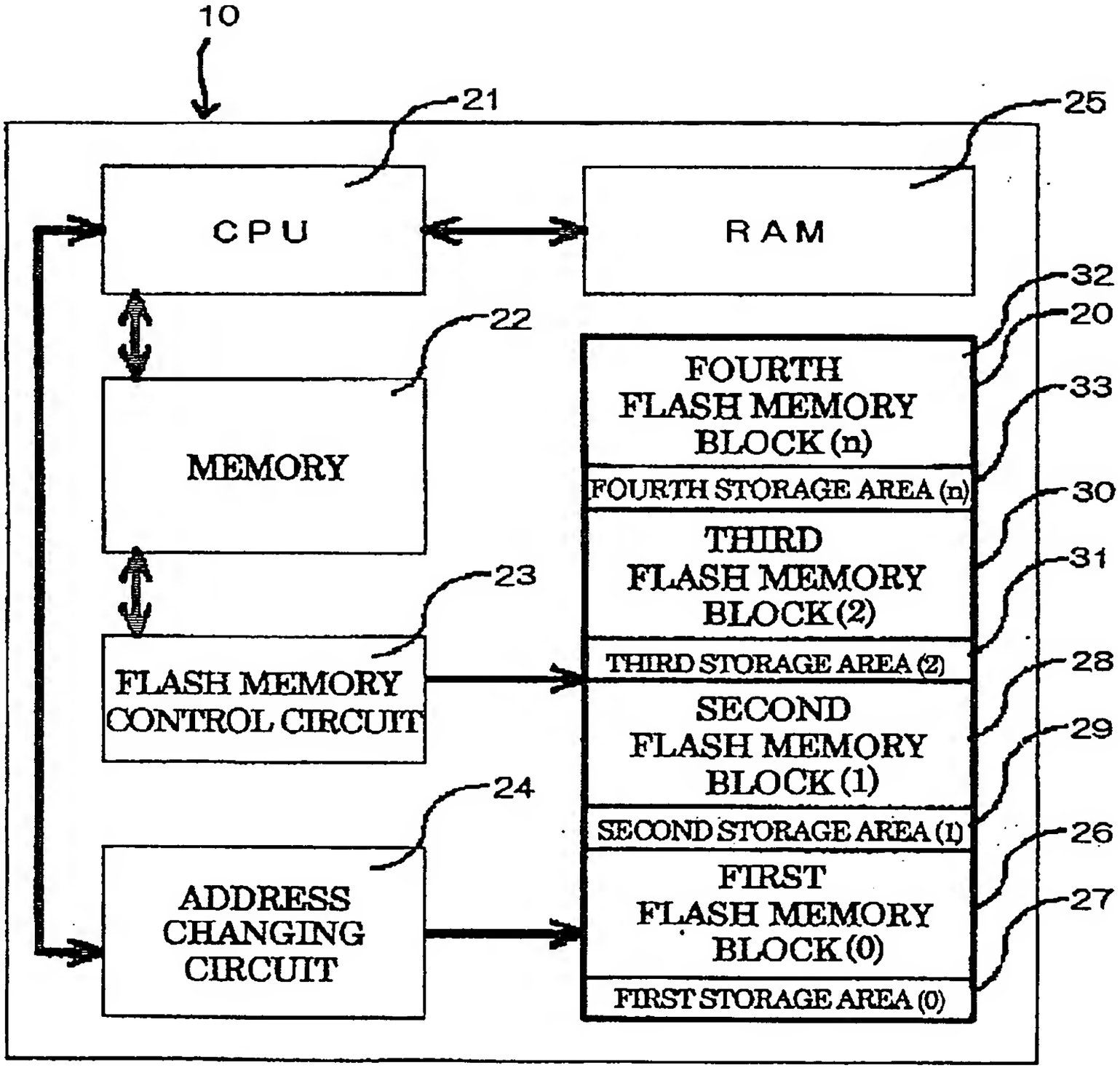

10
21
25
CPU
RAM
32
20
22
MEMORY
FOURTH FLASH MEMORY BLOCK (n)
33
FOURTH STORAGE AREA (n)
30
THIRD FLASH MEMORY BLOCK (2)
31
23
FLASH MEMORY CONTROL CIRCUIT
THIRD STORAGE AREA (2)
28
SECOND FLASH MEMORY BLOCK (1)
29
24
SECOND STORAGE AREA (1)
26
ADDRESS CHANGING CIRCUIT
FIRST FLASH MEMORY BLOCK (0)
27
FIRST STORAGE AREA (0)

FIG. 4

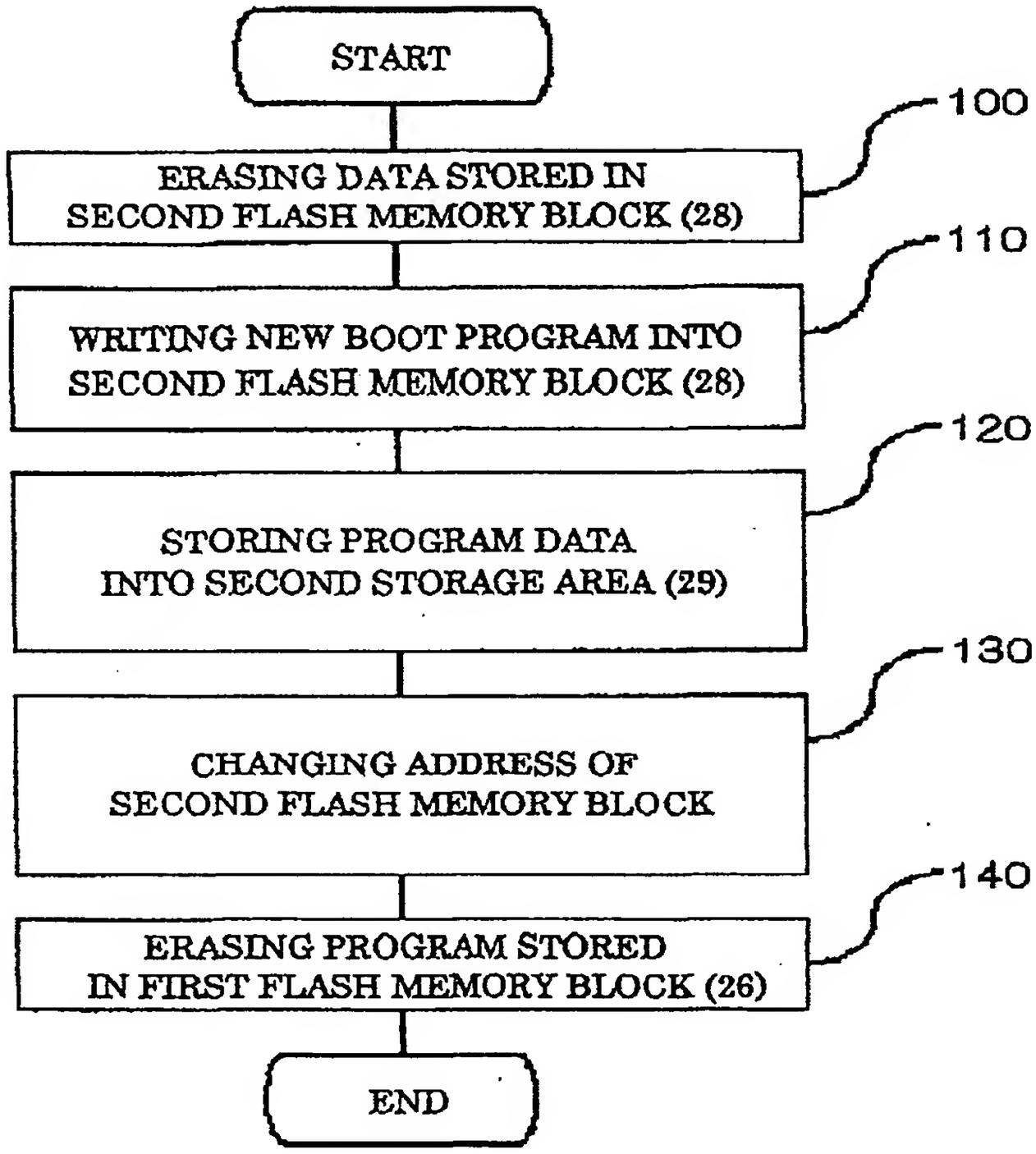

START
ERASING DATA STORED IN SECOND FLASH MEMORY BLOCK (28)
100
WRITING NEW BOOT PROGRAM INTO SECOND FLASH MEMORY BLOCK (28)
110
STORING PROGRAM DATA INTO SECOND STORAGE AREA (29)
120
CHANGING ADDRESS OF SECOND FLASH MEMORY BLOCK
130
ERASING PROGRAM STORED IN FIRST FLASH MEMORY BLOCK (26)
140
END

FIG. 5

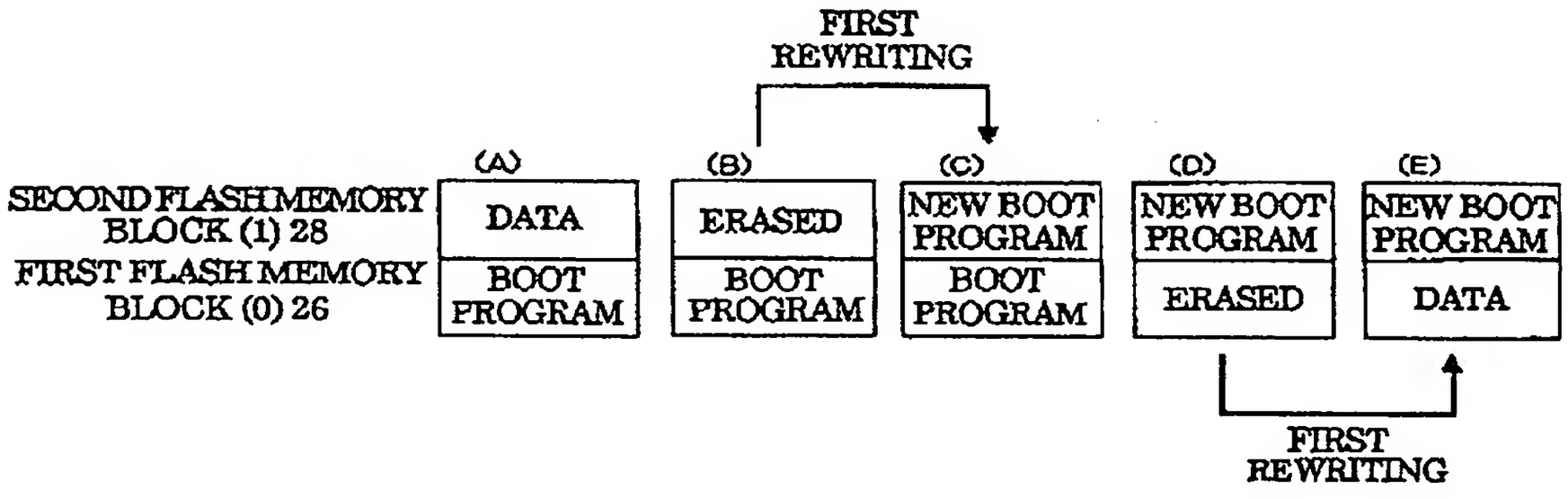

FIRST
REWRITING
(A)
(B)
(C)
(D)
(E)
SECOND FLASH MEMORY
BLOCK (1) 28
FIRST FLASH MEMORY
BLOCK (0) 26
DATA
BOOT
PROGRAM
ERASED
BOOT
PROGRAM
NEW BOOT
PROGRAM
BOOT
PROGRAM
NEW BOOT
PROGRAM
ERASED
NEW BOOT
PROGRAM
DATA
FIRST
REWRITING

FIG. 6

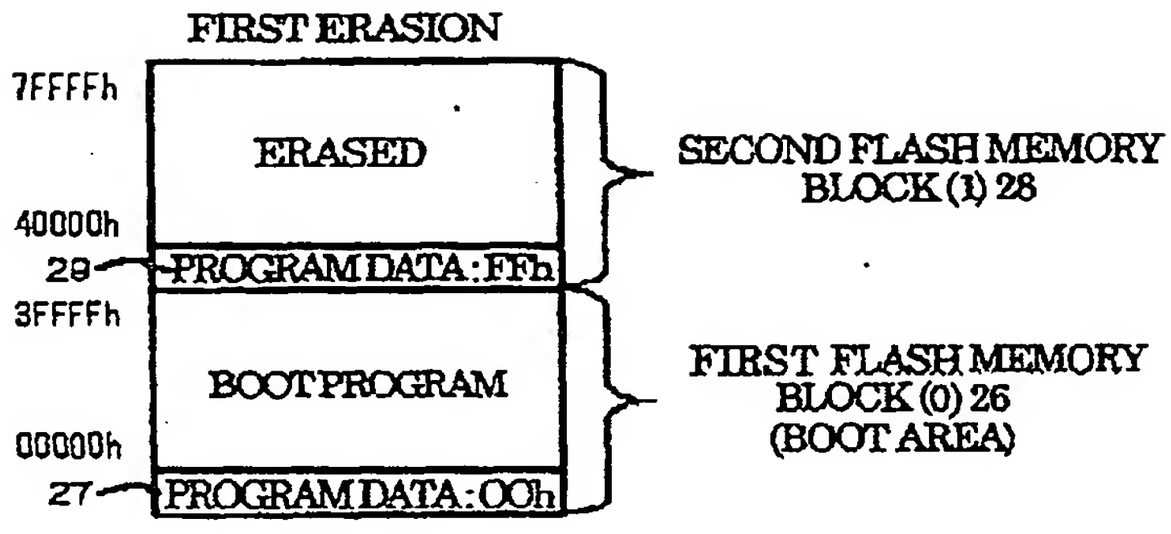

FIRST ERASION
7FFFFh
ERASED
40000h
28
PROGRAM DATA : FFh
3FFFFh
BOOT PROGRAM
00000h
27
PROGRAM DATA : 00h
SECOND FLASH MEMORY BLOCK (1) 28
FIRST FLASH MEMORY BLOCK (0) 26 (BOOT AREA)

FIG. 7

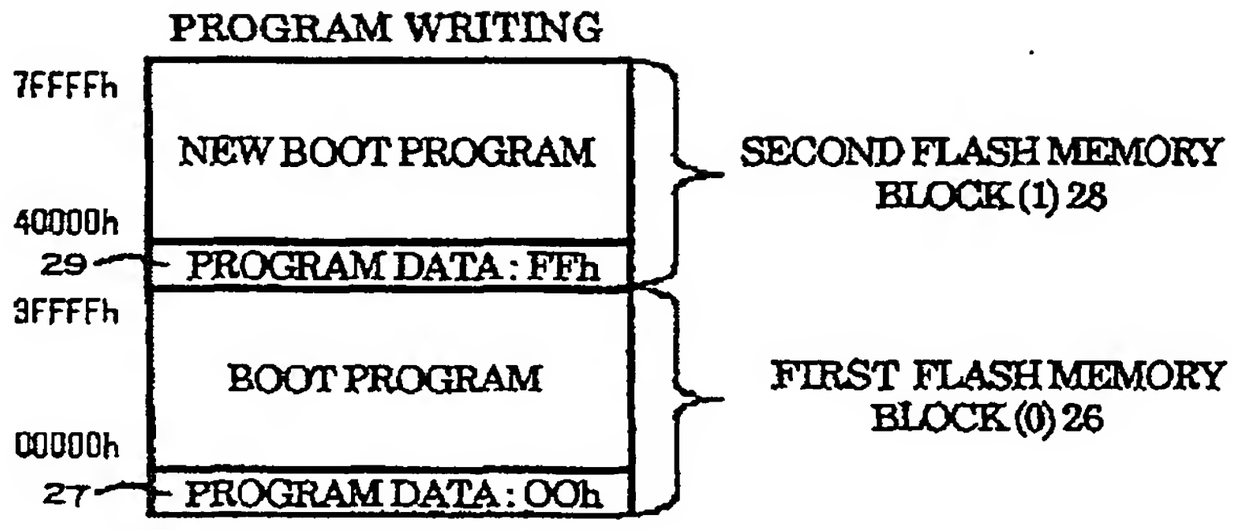

PROGRAM WRITING
7FFFFh
NEW BOOT PROGRAM
40000h
29
PROGRAM DATA : FFh
3FFFFh
BOOT PROGRAM
00000h
27
PROGRAM DATA : 00h
SECOND FLASH MEMORY BLOCK (1) 28
FIRST FLASH MEMORY BLOCK (0) 26

FIG. 8

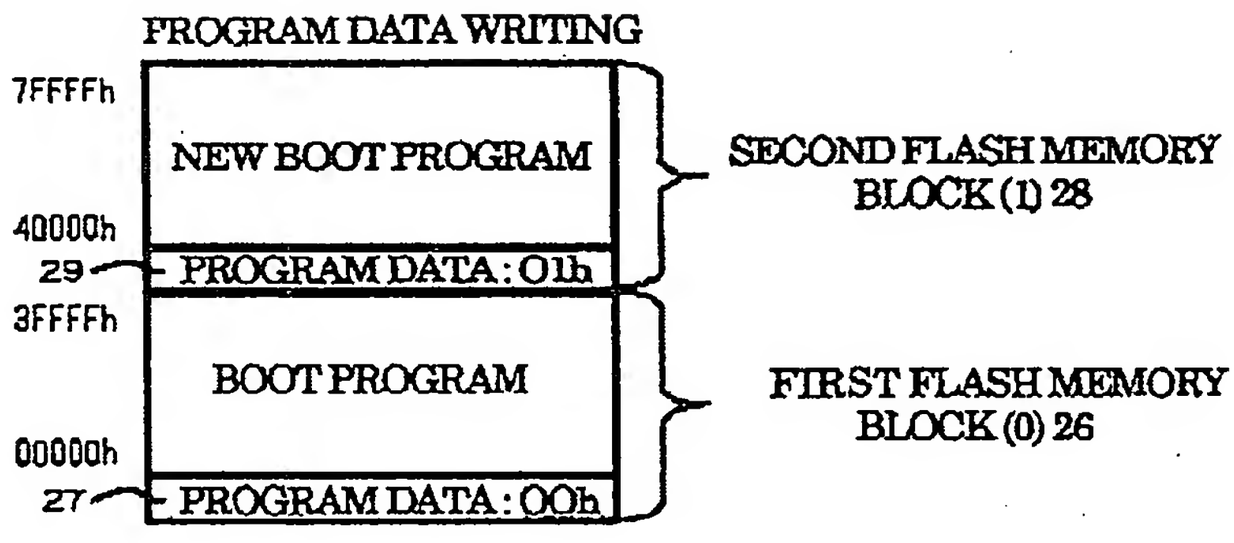

PROGRAM DATA WRITING
7FFFFh
NEW BOOT PROGRAM
40000h
29
PROGRAM DATA : 01h
3FFFFh
BOOT PROGRAM
00000h
27
PROGRAM DATA : 00h
SECOND FLASH MEMORY BLOCK (1) 28
FIRST FLASH MEMORY BLOCK (0) 26

FIG. 9

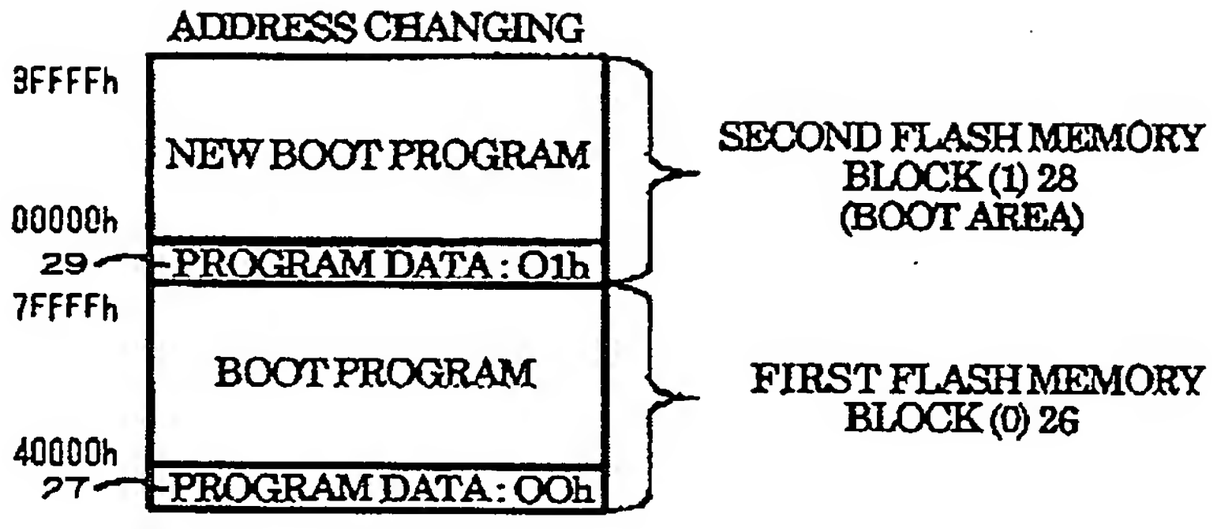

ADDRESS CHANGING
3FFFFh
NEW BOOT PROGRAM
00000h
29
PROGRAM DATA : 01h
7FFFFh
BOOT PROGRAM
40000h
27
PROGRAM DATA : 00h
SECOND FLASH MEMORY BLOCK (1) 28 (BOOT AREA)
FIRST FLASH MEMORY BLOCK (0) 26

FIG. 10

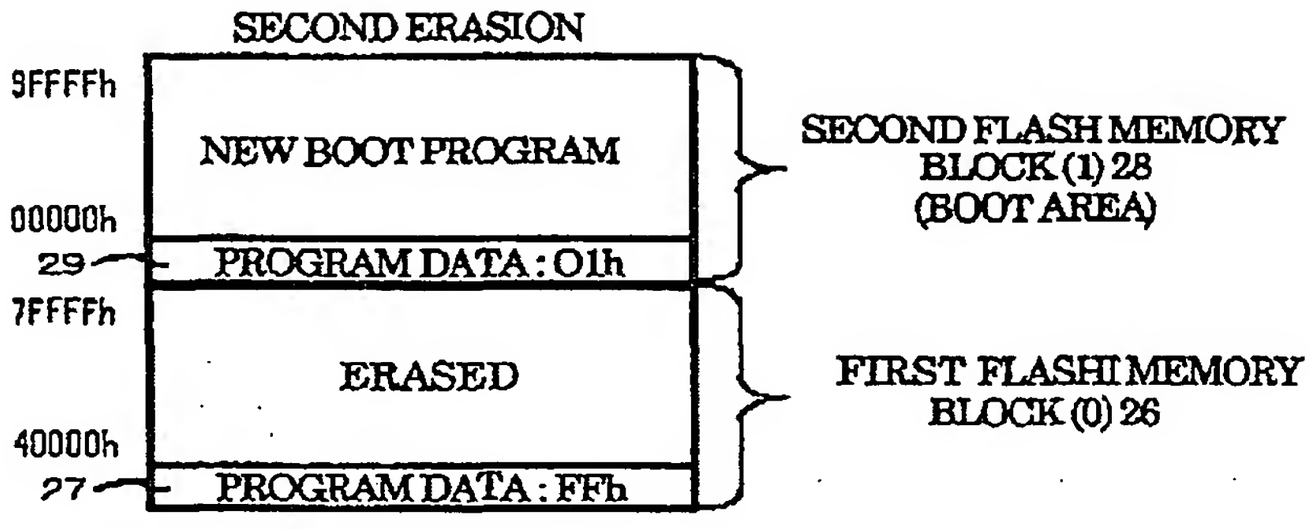

SECOND ERASION
9FFFFh
NEW BOOT PROGRAM
00000h
29
PROGRAM DATA : 01h
7FFFFh
ERASED
40000h
27
PROGRAM DATA : FFh
SECOND FLASH MEMORY
BLOCK (1) 28
(BOOT AREA)
FIRST FLASHI MEMORY
BLOCK (0) 26

FIG. 11

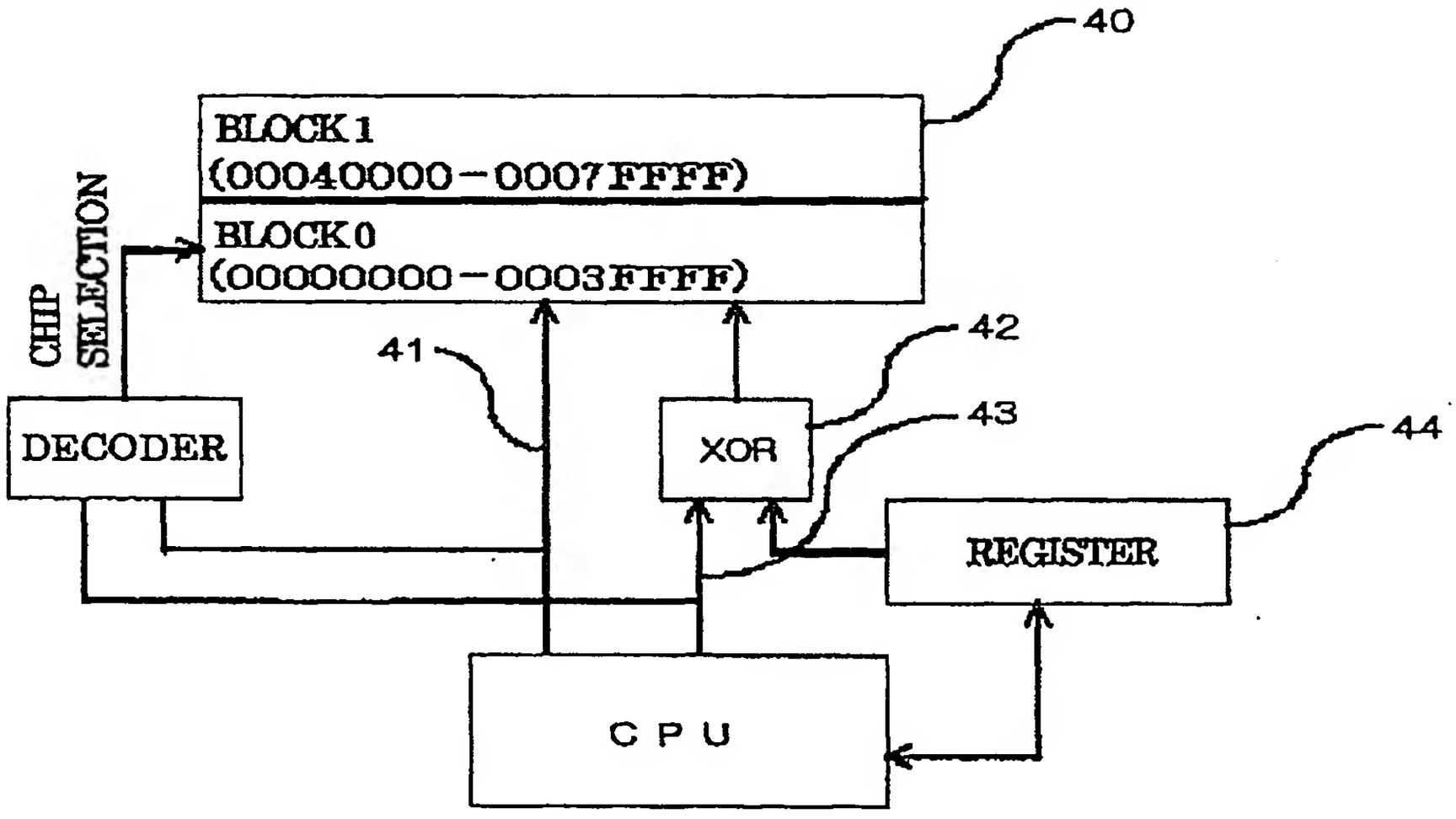
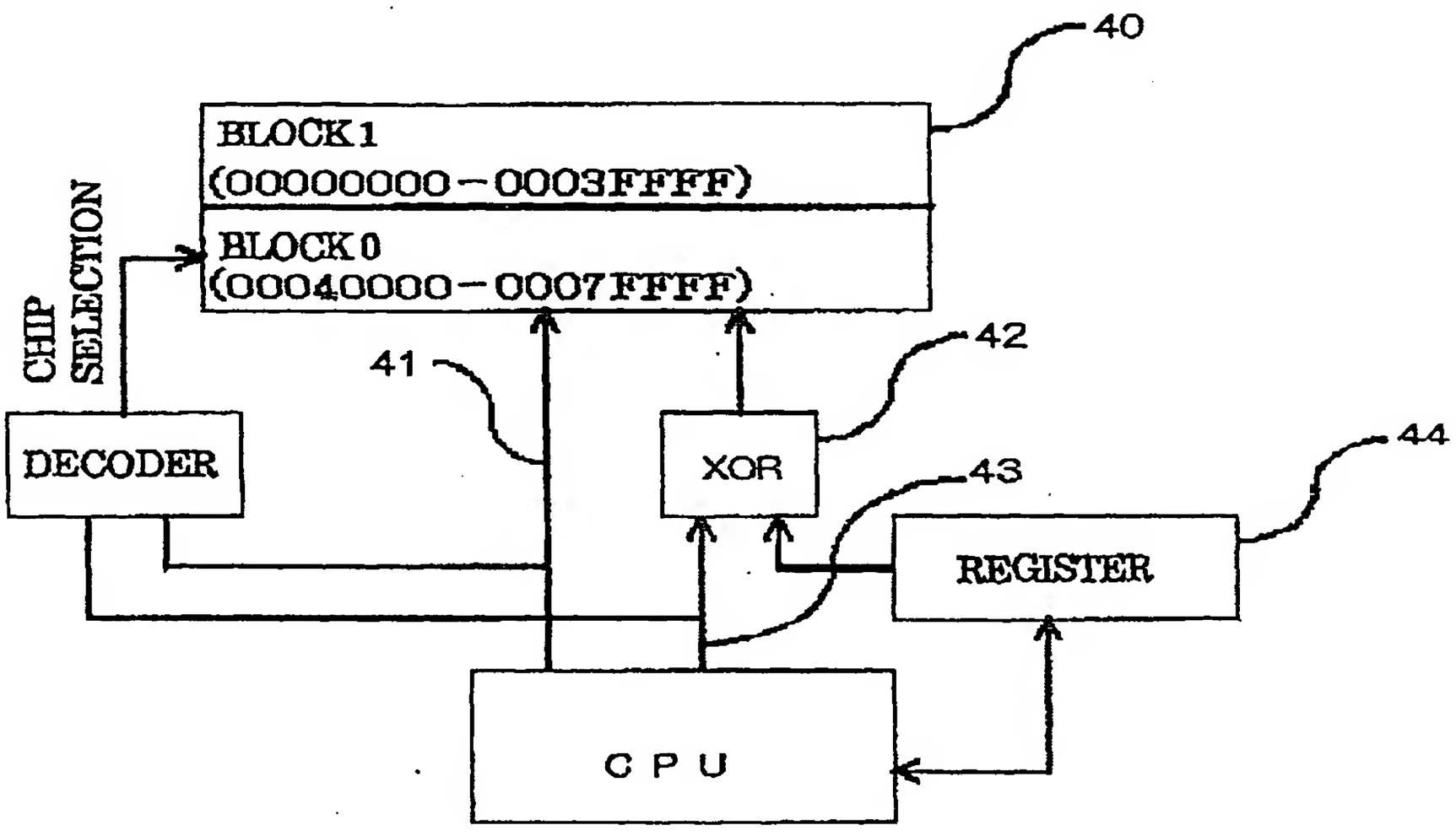

40
BLOCK 1
(00040000－0007FFFF)
BLOCK 0
(00000000－0003FFFF)
CHIP SELECTION
41
42
43
44
DECODER
XOR
REGISTER
C P U

FIG. 12


40
BLOCK1
(00000000-0003FFFF)
BLOCK0
(00040000-0007FFFF)
CHIP SELECTION
41
42
44
43
DECODER
XOR
REGISTER
CPU

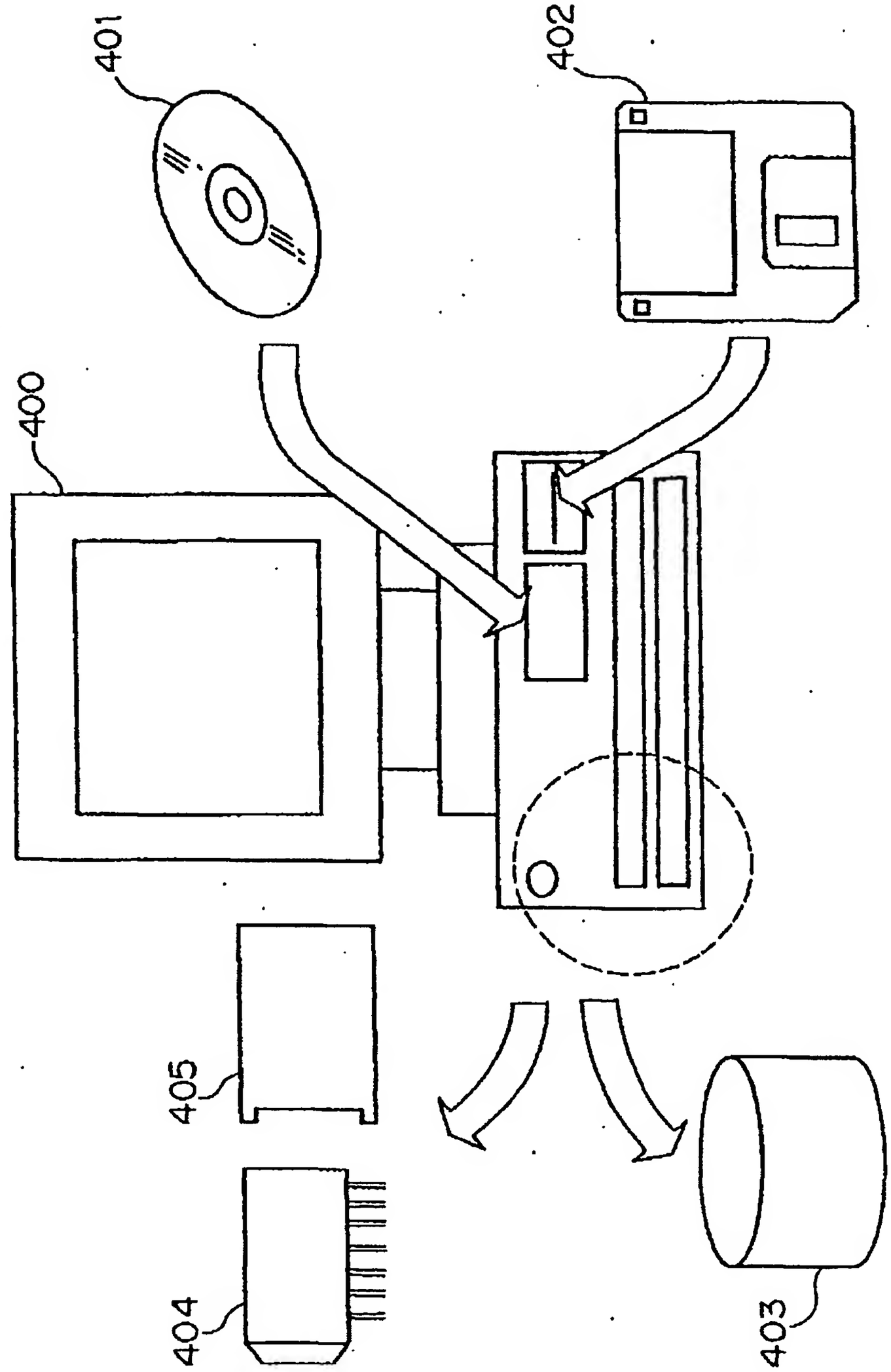


FIG. 13